PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-162013
(43)Date of publication of application : 19.06.1998

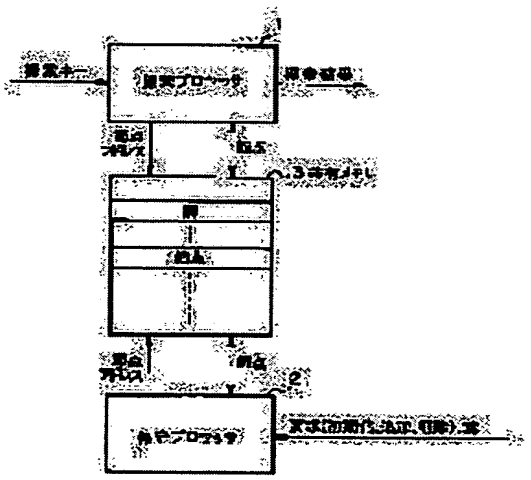
(51)Int.Cl. G06F 17/30

(21)Application number : 08-317918
(22)Date of filing : 28.11.1996

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
(72)Inventor : TAKAHASHI NAOHISA
MARUYAMA MITSURU
SANEI TAKESHI
OGURA TAKESHI
KAWANO TETSUO
YAGI SATORU

(54) DIGITAL SEARCHING DEVICE

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a digital searching device which executes a searching processing on a digital search tree at a high speed even if the number of headers in the digital search tree is increased, even if the request frequency of the searching processing, an elimination processing and an addition processing increase or even if the requests of the searching processing are continuously outputted.
SOLUTION: In the digital searching device, the digital search tree is kept in a common memory 3 and a maintenance processor 2 executes the initialization processing of the digital search tree, the addition processing of leaves and nodes and the elimination processing of the leaves and the nodes. A searching processor 1 executes the searching processing of the digital search tree. The maintenance processor 2 and the searching processor 1 independently operate and they access to the common memory 3 so as to operate the digital search tree in parallel.



LEGAL STATUS

[Date of request for examination]	21.12.1998
[Date of sending the examiner's decision of rejection]	21.11.2001
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3284064
[Date of registration]	01.03.2002
[Number of appeal against examiner's decision of rejection]	2001-22980
[Date of requesting appeal against examiner's decision of rejection]	21.12.2001
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（A）**　(11)特許出願公開番号

特開平11－73329

(43)公開日　平成11年(1999)３月16日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/45		Ｇ０６Ｆ　9/44	３２２Ｈ
12/08		12/08	Ｓ

審査請求　未請求　請求項の数６　ＯＬ　（全 23 頁）

(21)出願番号　特願平10－13878

(22)出願日　平成10年(1998)１月27日

(31)優先権主張番号　特願平9－166777

(32)優先日　平９(1997)６月24日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　津幡　真太郎
大阪府門真市大字門真1006番地　松下電器産業株式会社内

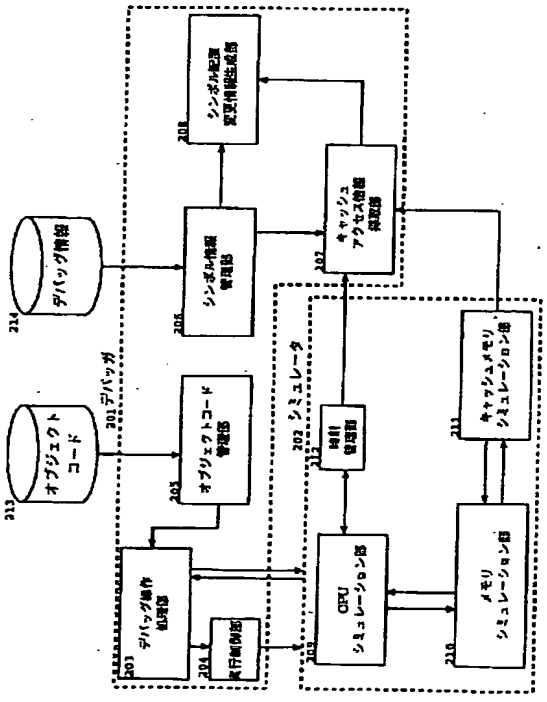
(74)代理人　弁理士　宮井　暎夫

(54)【発明の名称】　ソフトウエア開発支援システム

(57)【要約】

【課題】キャッシュメモリへのアクセスの状況を高級言語レベルで検出し、キャッシュメモリのヒット率が向上するようにシンボルの割付情報を生成することにより、キャッシュメモリ搭載プロセッサ上のプログラム開発を支援するソフトウエア開発支援システムを提供する。

【解決手段】　プロセッサの動作をシミュレーションするシミュレータ２０２と、キャッシュメモリへのアクセスに関する情報を採取するキャッシュアクセス情報採取部２０７と、プログラムのシンボル情報を管理するシンボル情報管理部２０６と、プログラムのシンボルの配置を変更するためのシンボル配置変更情報を生成するシンボル配置変更情報生成部２０８とを備えている。



1

【特許請求の範囲】

【請求項１】　キャッシュメモリを搭載するプロセッサ上で実行されるプログラムのソフトウエア開発支援システムであって、

前記プロセッサの動作をシミュレーションするシミュレーション手段と、このシミュレーション手段から前記キャッシュメモリへのアクセスに関する情報を採取するキャッシュアクセス情報採取手段と、前記プログラムのシンボル情報を管理するシンボル情報管理手段と、前記プログラムのシンボルの配置を変更するためのシンボル配置変更情報を生成するシンボル配置変更情報生成手段とを備えたソフトウエア開発支援システム。

【請求項２】　キャッシュアクセス情報採取手段は、キャッシュメモリでのエントリの競合性ミスに関する情報を採取する請求項１記載のソフトウエア開発支援システム。

【請求項３】　シンボル配置変更情報生成手段は、キャッシュメモリでのエントリの競合性ミスを低減するための情報を生成する請求項１または請求項２記載のソフトウエア開発支援システム。

【請求項４】　キャッシュメモリを搭載するプロセッサ上で実行されるプログラムのソフトウエア開発支援システムであって、

前記プロセッサの動作をシミュレーションするシミュレーション手段と、このシミュレーション手段から前記キャッシュメモリへのアクセスに関する情報を採取するキャッシュアクセス情報採取手段と、前記プログラムのシンボル情報を管理するシンボル情報管理手段と、前記シンボル情報を用いて前記キャッシュアクセスの情報を表示する表示手段とを備えたソフトウエア開発支援システム。

【請求項５】　キャッシュアクセス情報採取手段は、アクセス対象のアドレスおよびアクセス時のプログラムカウンタ値、スタックポインタ値ならびにアクセスがヒットかどうかの情報を採取する請求項４記載のソフトウエア開発支援システム。

【請求項６】　表示手段は、キャッシュメモリでのエントリ別に分類して表示する請求項４または請求項５記載のソフトウエア開発支援システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、キャッシュメモリを搭載するプロセッサ上で実行されるプログラムの開発を支援するソフトウエア開発支援システムに関するものである。

【０００２】

【従来の技術】近年の電子技術の発展により、家電機器などの組み込み機器においてもマイクロコンピュータなどプロセッサを組み込み、プログラム開発を伴った機器開発が行われている。加えて、組み込み機器の高機能化

2

に伴い、プログラムが大規模になり、プログラムの開発および保守の効率化のため、Ｃ言語など高級言語によるプログラム開発が行われている。

【０００３】また、プログラムの大規模化に従い、大容量のメモリがプログラムのコード格納やデータ格納のために必要とされている。さらに、プログラムの処理の高速化のため、メモリアクセスを高速にするキャッシュメモリ搭載のプロセッサが組み込み機器に用いられている。図３に、キャッシュメモリ搭載のプロセッサのシステム構成の一例を示す。図３において、点線で囲まれた部分がプロセッサであり、６０１はＣＰＵ（中央演算装置）であり各種演算を行う。６０２は、プロセッサの外部に存在する主記憶装置であり、プログラム実行コード（オブジェクトコード）およびデータを記憶する。６０３はキャッシュメモリであり、主記憶装置へのアクセスを高速化するために機能する。６０４はバスであり、このバスを用いて、ＣＰＵ６０１、主記憶装置６０２、キャッシュメモリ６０３の間でデータが転送される。

【０００４】このようなプロセッサにおける、キャッシュメモリの機能について説明する。主記憶装置６０２上のアドレスに対する読込みアクセスの場合には、そのアドレスおよびそのデータがキャッシュメモリ６０３の対応するアドレス上に登録されていれば（キャッシュヒットと呼ぶ）、そのキャッシュメモリ６０３上のデータが参照され、そのアクセスが高速に実行される。

【０００５】また、アドレスおよびデータがキャッシュメモリ６０３上に未登録であれば（キャッシュミス）、主記憶装置６０２上のデータが読込まれると同時に、アドレスとデータがキャッシュメモリ６０３上に登録される。この場合のアクセス時間は、はキャッシュヒット時に比べて大きい。同様に、主記憶装置６０２上のアドレスに対する書きこみアクセスの場合は、そのアドレスおよびそのデータがキャッシュメモリ６０３の対応するアドレス上に登録されていれば（キャッシュヒット）、そのキャッシュメモリ６０３上のデータのみが更新されることにより、そのアクセスが高速に実行される。

【０００６】また、そのアドレスおよびデータがキャッシュメモリ６０３上に未登録であれば（キャッシュミス）、主記憶装置６０２上のデータおよびそのアドレスとデータがキャッシュメモリ６０３上に登録されると同時にキャッシュメモリ６０３上のデータのみが更新される。この場合のアクセス時間は、キャッシュヒット時に比べて大きい。

【０００７】なお、主記憶装置上のアドレスとキャッシュメモリ上への対応づけの方式には、主記憶装置上の１つのアドレスに１つのキャッシュメモリ上のアドレスが対応づくダイレクトマップ方式や、２つのキャッシュメモリ上のアドレスが対応づく２ウエイセットアソシエイティブ方式など幾つかの方式がある。また、キャッシュメモリの方式によってキャッシュメモリへの書きこみと

同時につねに主記憶装置上へも書きこみが行われる方式（ライトスルー方式）もある。

【０００８】以上説明したようなキャッシュメモリ６０３を搭載したプロセッサにおいて、実行されるプログラムの性能評価および性能向上は、プログラム実行時のキャッシュヒット、キャッシュミスの状況を考慮し行う必要がある。加えて、組み込み機器開発においては、リアルタイム応用など高速処理実現と低コスト実現の両立が求められ、より正確な性能評価およびそれに基いた性能向上が求められる。

【０００９】しかし、従来は、性能評価として、実機での実行時間の測定や、またシミュレーションによるサイクル数の見積もりなどが行われてきた。また、キャッシュ搭載のプロセッサを対象とする場合は、性能評価の指標としてキャッシュのヒット率つまりアクセス回数に対するキャッシュヒットの比率が専ら用いられてきた。

【００１０】このような性能評価を行い、速度要求を満していないプログラムのサブルーチンなど特定部分については、利用者がプログラムを変更し、速度の改善を図ってきた。プログラム変更の一つの方法は、キャッシュヒット率を向上させ、メモリアクセスの速度を向上させることであり、例えば、キャッシュヒット率が向上するように、プログラムで使用するデータの割付けアドレスを変更することである。

【００１１】

【発明が解決しようとする課題】しかしながら、従来の方法においてキャッシュヒット率だけではその詳細な状況は判らず、また、シミュレーションにおけるプログラム動作のトレース結果では、トレースされる対象がアドレスだけであり、実際に利用者が変更する対象である高級言語レベルのプログラムには対応しておらず、性能向上のためのプログラム変更が困難であるという問題を有していた。

【００１２】この発明は、上記の課題に鑑み、キャッシュメモリへのアクセスの状況を高級言語レベルで検出し、キャッシュメモリのヒット率が向上するようにシンボルの割付情報を生成することにより、キャッシュメモリ搭載プロセッサ上のプログラム開発を支援するソフトウエア開発支援システムを提供することを目的とする。また、キャッシュミスの分類の１つとして、キャッシュメモリの１つのエントリに複数の主記憶上のブロック（アドレス）が対応するために起因する競合性ミスがある。このキャッシュミスは、複数の主記憶上のブロックが１つのエントリに対応づかないようにプログラムを変更してやれば解決できる可能性があるキャッシュミスである。

【００１３】この発明は、したがって上記の課題に鑑み、キャッシュミスのうち特に競合性ミスを検出し競合しているシンボルを競合しないようにシンボルの割付情報を生成することにより、キャッシュメモリ搭載プロセッサ上のプログラム開発を支援するソフトウエア開発支援システムを提供することを目的とする。さらにこの発明は、上記の課題に鑑み、キャッシュメモリへのアクセスの状況を高級言語レベルで利用者に表示し、キャッシュメモリ搭載プロセッサ上のプログラムを支援するソフトウエア開発支援システムを提供することを目的とする。

【００１４】

【課題を解決するための手段】請求項１記載のソフトウエア開発支援システムは、キャッシュメモリを搭載するプロセッサ上で実行されるプログラムのソフトウエア開発支援システムであって、プロセッサの動作をシミュレーションするシミュレーション手段と、このシミュレーション手段からキャッシュメモリへのアクセスに関する情報を採取するキャッシュアクセス情報採取手段と、プログラムのシンボル情報を管理するシンボル情報管理手段と、プログラムのシンボルの配置を変更するためのシンボル配置変更情報を生成するシンボル配置変更情報生成手段とを備えたものである。

【００１５】請求項１記載のソフトウエア開発支援システムによれば、キャッシュアクセス情報採取手段で採取された情報と、シンボル情報管理手段で管理されているシンボル情報から、プログラムのメモリアクセス時の速度が高速になるようにシンボルの配置を変更するためのシンボルの配置変更情報を生成することができる。これにより、キャッシュメモリへのアクセスの状況が対象プログラムが記述される高級言語レベルで検知可能となり、アクセス速度を低下させるシンボルの配置を変更し、プログラムの実行性能を向上させることが可能となる。

【００１６】請求項２記載のソフトウエア支援開発システムは、請求項１において、キャッシュアクセス情報採取手段が、キャッシュメモリでのエントリの競合性ミスに関する情報を採取するものである。請求項２記載のソフトウエア支援開発システムによれば、請求項１と同様な効果のほか、キャッシュアクセス情報採取手段で採取された情報からキャッシュ競合性ミスを検出し、シンボル情報と関連づけて、シンボルの配置変更情報をシンボル配置変更情報生成手段で生成することができる。これにより、キャッシュメモリ搭載のプログラムを開発する際に、キャッシュメモリへのアクセスの状況特に競合性ミスをシンボルレベルないし高級言語レベルで検出し、キャッシュメモリで競合性ミスが発生するシンボルの配置を変更する情報を生成するなど、その競合を解消するためにシンボルの配置を変更することが容易にでき、プログラム開発の効率化が図れる。

【００１７】請求項３記載のソフトウエア支援開発システムは、請求項１または請求項２において、シンボル配置変更情報生成手段が、キャッシュメモリでのエントリの競合性ミスを低減するための情報を生成するものであ

5

る。請求項３記載のソフトウエア支援開発システムによれば、請求項１または請求項２と同様な効果のほか、競合性のキャッシュミスが発生したシンボルをキャッシュされていない空のエントリに対応するアドレスに変更するための情報を生成するなど、キャッシュメモリの競合性ミスを低減させるようなシンボルの配置変更情報が生成でき、より一層のプログラムの性能向上を図れる。

【００１８】請求項４記載のソフトウエア支援開発システムは、キャッシュメモリを搭載するプロセッサ上で実行されるプログラムのソフトウエア開発支援システムであって、プロセッサの動作をシミュレーションするシミュレーション手段と、このシミュレーション手段からキャッシュメモリへのアクセスに関する情報を採取するキャッシュアクセス情報採取手段と、プログラムのシンボル情報を管理するシンボル情報管理手段と、シンボル情報を用いてキャッシュアクセスの情報を表示する表示手段とを備えたものである。

【００１９】請求項４記載のソフトウエア支援開発システムによれば、キャッシュアクセス情報採取手段で採取された情報と、シンボル情報管理手段で管理されているシンボル情報を表示手段により関連付けて表示することができるので、キャッシュメモリへのアクセスの状況が、対象プログラムが記述される高級言語レベルで表示可能となり、評価性能に基いたプログラムの変更が容易になる。したがって、キャッシュメモリ搭載のプログラムを開発し評価する際に、キャッシュメモリへのアクセス状況を高級言語レベルで知ることができ、プログラム開発の効率化が図れる。

【００２０】請求項５記載のソフトウエア支援開発システムは、請求項４において、キャッシュアクセス情報採取手段が、アクセス対象のアドレスおよびアクセス時のプログラムカウンタ値、スタックポインタ値ならびにアクセスがヒットかどうかの情報を採取するものである。請求項５記載のソフトウエア支援開発システムによれば、請求項４と同様な効果がある。

【００２１】請求項６記載のソフトウエア支援開発システムは、請求項４または請求項５において、表示手段が、キャッシュメモリでのエントリ別に分類して表示するものである。請求項６記載のソフトウエア支援開発システムによれば、請求項４または請求項５と同様な効果がある。

【００２２】

【発明の実施の形態】以下、この発明の実施の形態について説明する。この発明の第１の実施の形態を図１から図１６により説明する。図１は、この発明の第１の実施の形態におけるソフトウエア開発支援システムに適用されるシステムの一例であり、高級言語によるソフトウエア開発およびそのシミュレーション、ならびにデバッグに係る全体的なシステムである。

【００２３】図１において、１０５は、開発の対象となるアプリケーションを高級言語で記述したソースプログラムである。１０１はコンパイラであり、ソースプログラム１０５を入力し、対象とするプロセッサに対応したオブジェクトコード１０６と、デバッグ情報１０７を出力する。

【００２４】１０２は、ソフトウエア開発支援システムであり、デバッガ１０３と、対象プロセッサの動作をシミュレーションするシミュレーション手段であるシミュレータ１０４とから構成される。開発支援システム１０２は、オブジェクトコード１０６とデバッグ情報１０７を入力し、アプリケーションのシミュレーションおよび高級言語レベルでのデバッグを実現する。

【００２５】図９に、開発支援システム１０２が対象とするプロセッサの構成図を示す。図９は図３と比べて、キャッシュメモリが命令キャッシュメモリ１３０３とデータキャッシュメモリ１３０４に別れている点が異なる。よって、命令キャッシュメモリ１３０３とデータキャッシュメモリ１３０４についてのみ説明する。命令キャッシュメモリ１３０３は、命令の読込み動作（フェッチ動作）時に、用いられるキャッシュメモリであり、データキャッシュメモリ１３０４は、データの読込み動作（データリード動作）時とデータの書きこみ動作（データライト動作）時に用いられるキャッシュメモリである。

【００２６】また、命令キャッシュメモリ１３０３は、内部にタグアレイ１３０６およびデータアレイ１３０７を備え、同様にデータキャッシュメモリ１３０４は、タグアレイ１３０８とデータアレイ１３０９を備える。タグアレイ１３０６、１３０８は、主記憶装置１３０２上のアドレスとキャッシュメモリのデータアレイ１３０７、１３０９上のアドレスを対応付けるタグ情報を保持する。また、データアレイ１３０７、１３０９は、上記のように対応付けられたデータの内容自体を保持する。

【００２７】図４は、プロセッサのＣＰＵの構成とともに、メモリとの関係を示す。図４において、７０１はＣＰＵであり、プログラム実行時に主にデータを格納するデータレジスタ（Ｄ０、Ｄ１、Ｄ２、Ｄ３）７０３、７０４、７０５、７０６と、主にアドレスを格納するアドレスレジスタ（Ａ０、Ａ１、Ａ２、Ａ３）７０７、７０８、７０９、７１０と、演算結果フラグを格納するプロセッサ状態語レジスタ（ＰＳＷ）７１１と、プログラム実行アドレスを指し示すプログラムカウンタレジスタ（ＰＣ）と、スタック領域のアドレスを指し示すスタックポインタレジスタ（ＳＰ）と、各種演算を行う演算器７１４から構成される。

【００２８】７０２は、メモリ領域の概念図である。７０２に示すように、メモリ領域は、プログラム自体である実行コードを格納する実行コード格納領域７１５と、主にグローバルなデータを格納するデータ領域７１６と、主にローカルなデータを格納するスタック領域７１

6

7

７から構成される。このメモリ領域には、すべてアドレスが割振られ、ＣＰＵ７０１からはアドレスによりアクセスが行われる。　この構成において、プロセッサは、実行コード格納領域７１５に格納されている命令を読込み、読込んだ命令に従って各種レジスタあるいはメモリ上のデータを用いて演算を行う。

【００２９】このようなプロセッサでは、プロセッサ（ＣＰＵ）とメモリ（主記憶）の間のデータの転送は、（１）メモリからプロセッサへの命令の読込み（フェッチ）、（２）メモリからプロセッサへのデータの読込み（データリード）、（３）プロセッサからメモリへのデータの書きこみ（データライト）の３種類に区分される。

【００３０】表１に、ソースプログラムの一例を示す。

【００３１】

【表１】

```
01:  int x, y;
02:
03:  void main() {
04:    int i;
05:    x = y = 0;
06:    for (i=0; i<100; i++) {
07:      x = i*i - i;
08:      y = y + x*2;
09:    }
10:  }
```

8

＊【００３２】表１に示すソースプログラムは、高級言語の一つであるＣ言語によって記述されている。なお、表１において、各行の左端の『数字：』は、説明のために付与した行番号である。表１のソースプログラムにおいて、１行目は、グローバル変数でｉｎｔ（整数）型変数ｘとｙの宣言を示す。３行目から１０行目までが、関数ｍａｉｎの定義部分である。関数ｍａｉｎでは、４行目に示すように、ローカル変数でｉｎｔ（整数）型変数ｉを用いて、６行目から９行目に示すように、変数ｉを０から９９まで増加させて、７、８行目に示す演算を繰り返し行う処理が記述されている。

【００３３】表２に、オブジェクトコードに対応するアセンブラ記述による説明図を示す。オブジェクトコードは、対象とするプロセッサの機械語命令で表現され、アドレスとそのアドレスに格納する値が情報として記述される。

【００３４】

【表２】

```
01:  _TEXT    SECTION
02:
03:  0x100          add     -4,SP
04:  0x104          clr     D0
05:  0x108          mov     D0,(0x410)
06:  0x10C          mov     D0,(0x400)
07:  0x110          mov     D0,(SP)
08:  0x114          jmp     0x150
09:
10:  0x118          mov     (SP),D0
11:  0x11C          mov     D0,D1
12:  0x120          mul     D1,D0
13:  0x124          mov     (SP),D1
14:  0x128          sub     D1,D0
15:  0x12C          mov     D0,(0x400)
16:  0x130          mov     D0,D1
17:  0x134          mov     (0x410),D0
18:  0x138          add     D1,D0
19:  0x13C          add     D1,D0
20:  0x140          mov     D0,(0x410)
21:  0x144          mov     (SP),D0
22:  0x148          inc     D0
23:  0x14C          mov     D0,(SP)
24:
25:  0x150          mov     (SP),D0
26:  0x154          cmp     100,D0
27:  0x158          bge     0x160
28:  0x15C          jmp     0x118
29:
30:  0x160          ret
31:
32:
33:  _DATA    SECTION
34:  0x400          DS      4
35:  0x410          DS      4
```

【００３５】表２では、説明のため機械語命令と対応するアセンブラ記述を示しているが、本質的には前記の２つは同一の情報を表現する。また、全ての機械語命令は、４バイトの長さのビットパターンである。なお、表２において、各行の左端の『数字：』は、説明のために付与した行番号であり、左端２番目の１６進数による表記は、その命令あるいはデータが格納されるアドレスを示す。例えば、３行目の命令『ａｄｄ　－４，ＳＰ』に対応するオブジェクトコードは、アドレス０ｘ１００番地に格納され、３３行目に示すデータｘに対応するデー

タは、アドレス０ｘ４００番地に格納される。

【００３６】表２において、１行目から３０行目までは、実行コード部分である。１行目は、アセンブラ記述での擬似命令であり、１行目に続く部分が実行コードであることを示す。また、３３行目から３５行目は、データ領域部分であり、３３行目は、以下に続く部分がデータ領域であることを示すアセンブラ擬似命令である。３４行目は、アドレス０ｘ４００番地以降に４バイト分の領域を割当ることを示す。同様に、３５行目は、アドレス０ｘ４１０番地以降に４バイト分の領域を割当ることを示す。

【００３７】後述するシンボル情報によって、アドレス０ｘ１００番地がＣ言語プログラム上の関数ｍａｉｎの先頭番地に対応付けられ、また、アドレス０ｘ４００番地が変数ｘに、同様にアドレス０ｘ４１０番地が変数ｙに対応付けられる。下記に３行目から３０行目に記述されている実行コードで用いられているアセンブラ記述とプロセッサの命令の対応を示す。

【００３８】（１）ａｄｄ　『加算データ』，『被加算データ』

『加算データ』で示されるレジスタあるいはメモリの内容あるいは即値を『被加算データ』で示されるレジスタの内容に加算して、結果を『被加算データ』で示されるレジスタに格納する。例えば、３行目の記述は、即値－４をスタックポインタレジスタ（ＳＰ）の内容に加算して、加算結果をスタックポインタレジスタに格納する命令である。また、１８行目の記述は、データレジスタＤ０の内容をデータレジスタＤ１の内容に加算して、加算結果をデータレジスタＤ１へ格納する命令である。

【００３９】（２）ｃｌｒ　『対象レジスタ』

『対象レジスタ』で示されるレジスタの内容を値０にする。例えば、４行目の記述は、データレジスタＤ０の内容を０にする命令である。

（３）ｉｎｃ　『対象レジスタ』

『対象レジスタ』で示されるレジスタの内容を１増加して格納する。例えば、２２行目の記述は、データレジスタＤ０の内容を１増加して格納する命令である。

【００４０】（３）ｓｕｂ　『減算データ』，『被減算データ』

『減算データ』で示されるレジスタあるいはメモリの内容あるいは即値を『被減算データ』で示されるレジスタの内容から減算して、結果を『被減算データ』で示されるレジスタに格納する。例えば、１４行目の記述は、データレジスタＤ１の内容をデータレジスタＤ０の内容から減算して、減算結果をデータレジスタＤ０に格納する命令である。

【００４１】（４）ｃｍｐ　『減算データ』，『被減算データ』

『減算データ』で示されるレジスタあるいはメモリの内容あるいは即値を『被減算データ』で示されるレジスタ

の内容から減算して、結果のフラグ（結果が正である、結果が負である、結果が０である）をプロセッサ状態語レジスタＰＳＷに反映する。ｓｕｂ命令と異なり、減算結果は、レジスタには格納されない。例えば、２６行目の記述は、データレジスタＤ０から即値１００を減算して、その結果フラグをプロセッサ状態語レジスタＰＳＷに反映する命令である。

【００４２】（５）ｍｏｖ　『転送元データ』，『転送先データ』

『転送元データ』で示されるレジスタあるいはメモリの内容あるいは即値を『転送先データ』で示されるレジスタあるいはメモリへ転送する。例えば、５行目の記述は、データレジスタＤ０の内容をアドレス０ｘ４１０番地から始まる１ワード（４バイト）分のメモリ上の領域に転送する命令である。また、例えば、１０行目の記述は、スタックポインタレジスタＳＰの内容が指し示すメモリ上の１ワード分の領域の内容を、データレジスタＤ０へ格納する命令である。

【００４３】（６）ｊｍｐ　『ジャンプ先』

『ジャンプ先』で示されるアドレスへ、プログラム実行の制御を変更する。例えば、８行目の記述は、０ｘ１５０番地（２５行目）にプログラム実行の制御を変更する命令である。なお、このようなジャンプ命令以外は、プログラム実行は逐次的に行われる。例えば、０ｘ１００番地（３行目）の命令の次は０ｘ１０４番地（４行目）の命令が実行される。

【００４４】（７）ｍｕｌ『乗算データ』，『被乗算データ』

『乗算データ』で示されるレジスタの内容を、『被乗算データ』で示されるレジスタの内容に乗じて、乗算結果を『被乗算データ』で示されるレジスタに格納する。例えば、１２行目の記述は、データレジスタＤ１の内容をデータレジスタＤ０の内容に乗算して、その乗算結果をデータレジスタＤ０に格納する命令である。

【００４５】（８）ｂｇｅ　『ジャンプ先』

プロセッサ状態語レジスタのフラグが正あるいは０の場合に、『ジャンプ先』で示されるアドレスへ、プログラム実行の制御を変更する。例えば、２７行目の記述は、フラグが正あるいは０の場合に、０ｘ１６０番地（３０行目）にプログラム実行の制御を変更する命令である。

【００４６】（９）ｒｅｔ

サブルーチン呼び出し元へプログラムの制御を変更する。しかし、オブジェクトコードには、前記に説明したようなＣ言語での関数あるいは変数との対応付けの情報はなく、実際には次に説明するシンボル情報で示される。表３に、デバッグ情報の一例であるシンボル情報を示す。

【００４７】

【表３】

11

シンボル情報

```
01:  関数 main
02:       開始アドレス   0x100
03:       引　数          void
04:       返り値          void
05:       ローカル変数
06:         int i          (SP)
07:
08:
09:  変数 x
10:       開始アドレス   0x400
11:       int型
12:
13:  変数 y
14:       開始アドレス   0x410
15:       int型
```

【００４８】なお、表３において、各行の左端の『数字：』は、説明のために付与した行番号である。シンボル情報では、高級言語上の変数名や関数名などシンボルとオブジェクトコード上でのアドレスとの対応付けを行う情報である。表３において、１行目から６行目では、シンボル″ｍａｉｎ″が関数であり、その開始アドレスが０ｘ１００番地であり、関数の引数および戻り値がｖｏｉｄ型であり、加えて関数ｍａｉｎ内のローカル変数ｉがスタックポインタ（ＳＰ）レジスタが指すアドレスに割付られており、その型がｉｎｔ（整数）型であることを示している。

【００４９】また、９行目から１１行目までには、シンボル″ｘ″がグローバル変数であり、その開始アドレスが０ｘ４００番地であり、その型がｉｎｔ（整数）型であることを示している。同様に、１３行目から１５行目では、シンボル″ｙ″がグローバル変数であり、その開始アドレスが０ｘ４１０番地であり、その型がｉｎｔ（整数）型であることを示している。

【００５０】図２に、前記開発支援システム１０２の構成の一例を示す。図２において、２０１はデバッガ１０３であり、デバッグ操作処理部２０３、実行制御部２０４、オブジェクトコード管理部２０５、シンボル情報管理手段であるシンボル情報管理部２０６、キャッシュアクセス情報採取手段であるキャッシュアクセス情報採取部２０７、シンボル配置変更情報生成手段であるシンボル配置変更情報生成部２０８を備える。

【００５１】また、２０２はシミュレータ１０４であり、ＣＰＵをシミュレーションするＣＰＵシミュレーション部２０９、内蔵メモリや外付けメモリをシミュレーションするメモリシミュレーション部２１０、キャッシュメモリをシミュレーションするキャッシュメモリシミュレーション部２１１、シミュレーション実行の時刻を管理計測する時刻管理部２１２を備える。

【００５２】デバッグ操作処理部２０３は、オブジェクトコード２１３のロードや実行開始、ブレークアドレス設定などユーザからのデバッガコマンドの入力を受付け、処理する手段である。例えば、オブジェクトコード２１３のロードコマンド処理時には、オブジェクトコード管理部２０５から、オブジェクトコード２１３を読込み、シミュレータ２０２内のメモリシミュレーション部２１０にオブジェクトコード２１３の内容を書き込む処理を行う。

12

【００５３】また、実行開始コマンド処理時には、シミュレータ２０３内のＣＰＵシミュレーション部２０９のプログラムカウンタ（ＰＣ）レジスタに開始アドレスを設定し、実行制御部２０４により、オブジェクトコードのシミュレーションの実行を開始する。実行制御部２０４は、シミュレータ２０２を用いて対象オブジェクトコードの実行を実現する。シミュレーションの実行は、命令を単位として制御可能である。なお、これはプロセッサの動作サイクル毎の制御でもよい。

【００５４】この実行制御部２０４では、ユーザに指定されたブレークポイントにオブジェクトコードの実行が到達するまで、あるいは既定の停止アドレス（例えばｅｘｉｔ関数のアドレス）に実行が到達するまで、オブジェクトコード２１３の実行が繰り返し行われる。図５に、実行制御部２０４の動作フローチャートを示す。図５において、ステップ８０１では、シミュレータ２０２からＣＰＵ７０１（１３０１）内のプログラムカウンタレジスタ（ＰＣ）の値を読み出し、実行アドレスが既定の停止アドレス（例えばｅｘｉｔ関数）に到達したかを判定する。到達していれば処理を終了し、到達していなければステップ８０２に進む。

【００５５】ステップ８０２では、デバッグ操作処理部２０３から利用者が指定したブレークアドレスの情報を読み出し、実行アドレスがブレークアドレスに到達したかを判定する。到達していれば処理を終了し、到達していなければステップ８０３に進む。ステップ８０３では、シミュレーション部２０２を用いて１命令分のシミュレーションを行う。

【００５６】図２において、２０９はＣＰＵシミュレーション部であり、ＣＰＵ７０１（１３０１）の動作をシミュレーションする。つまり、実行する命令をプログラムカウンタ（ＰＣ）レジスタが指し示すアドレスから読取り（以下フェッチ動作とよぶ）、読込んだ命令に従ってレジスタあるいはメモリ上のデータを用いて演算を行う。

【００５７】フェッチ動作をシミュレーションする場合、メモリシミュレーション部２１０にアドレス（フェッチアドレス）を入力して、そのアドレス上のデータを得る。また、演算のシミュレーションにおいて、入力としてメモリ上のデータを読込む（データリード）際には、メモリシミュレーション部２１０にアドレスを入力し、そのアドレス上のデータを得る。

【００５８】同様に演算時に結果をメモリ上に出力する（データライト）際には、メモリシミュレーション部２

１０へアドレスと出力するデータ内容を入力し、そのアドレス上のデータへ書き込む。図６に、ＣＰＵシミュレーション部２０９の１命令分のシミュレーションを行う場合の動作フローチャートを示す。図６において、ステップ９０１では、ＣＰＵシミュレーション部２０９のプログラムカウンタ（ＰＣ）レジスタが指し示すアドレスと、１命令の語長である４バイトをメモリシミュレーション部２１０に指定して、そのアドレス上のデータを読み込む。また、同時に命令の読込みに必要なサイクル数を求める。

【００５９】ステップ９０２では、入力した機械語命令である４バイト長のビットパターンを解析し、命令の種類および命令実行に必要な入力先および出力先を求める。ビットパターンには、入力元あるいは出力先のレジスタの指定番号や、即値あるいはメモリ上のアドレスが埋込まれている。ステップ９０３では、命令実行に必要なデータを入力する。データの入力元は命令によって異なり、レジスタの内容やメモリ上の内容あるいは即値である。入力元がレジスタの内容である場合は、ＣＰＵシミュレーション部２０９の中のレジスタ内容管理領域を参照することによって、入力値を得る。また、入力元がメモリ上の内容の場合は、メモリシミュレーション部２１０に入力元アドレスおよびデータ長を指定することにより、入力値を得る。また、同時にデータの読込みに必要なサイクル数を求める。また、即値の場合は、ステップ９０２で解析した結果を参照する。

【００６０】ステップ９０４では、ステップ９０２で解析した結果を用いて命令の種類を判別し、命令の種類に応じてステップ９０５ａ、９０５ｂ、９０５ｃ、９０５ｄ、９０５ｅ、９０５ｆ、９０５ｇ、９０５ｈ、９０５ｉに進む。命令がａｄｄ命令の場合は、ステップ９０５ａに進む。ステップ９０５ａでは、ａｄｄ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータを用いて加算を実行し、演算結果を得る。

【００６１】同様に、命令がｉｎｃ命令の場合は、ステップ９０５ｂに進む。ステップ９０５ｂでは、ｉｎｃ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータと値１の加算を実行し、演算結果を得る。同様に、命令がｓｕｂ命令の場合は、ステップ９０５ｃに進む。ステップ９０５ｃでは、ｓｕｂ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータを用いて減算を実行し、演算結果を得る。

【００６２】同様に、命令がｃｍｐ命令の場合は、ステップ９０５ｄに進む。ステップ９０５ｄでは、ｃｍｐ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータによる減算を実行し、演算結果フラグを得る。同様に、命令がｍｕｌ命令の場合は、ステップ９０５ｅに進む。ステップ９０５ｅでは、ｍｕｌ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータを用いて乗算を実行し、演算結果を得る。

【００６３】同様に、命令がｃｌｒ命令の場合は、ステップ９０５ｆに進む。ステップ９０５ｆでは、ｃｌｒ命令の演算シミュレーションを行う。つまり演算結果として０を得る。同様に、命令がｍｏｖ命令の場合は、ステップ９０５ｇに進む。ステップ９０５ｇでは、ｍｏｖ命令の演算シミュレーションを行う。つまりステップ９０３で入力したデータを演算結果とする。

【００６４】同様に、命令がｊｍｐ命令の場合は、ステップ９０５ｈに進む。ステップ９０５ｈでは、ｊｍｐ命令の演算シミュレーションを行う。つまりステップ９０２での解析結果からジャンプ先のアドレスを入力して、ＣＰＵシミュレーション部２０９のプログラムカウンタレジスタに格納する。同様に、命令がｂｇｅ命令の場合は、ステップ９０５ｉに進む。ステップ９０５ｉでは、ｂｇｅ命令の演算シミュレーションを行う。つまりＣＰＵシミュレーション部２０９からプロセッサ状態語レジスタＰＳＷの値を読み出し、フラグを参照して、条件が成立している場合（演算結果フラグが正あるいは０）に、ステップ９０２での解析結果からジャンプ先のアドレスを入力して、ＣＰＵシミュレーション部２０９のプログラムカウンタレジスタに格納する。

【００６５】ステップ９０５ａ、９０５ｂ、９０５ｃ、９０５ｄ、９０５ｅ、９０５ｆ、９０５ｇでは、次に処理終了後ステップ９０６に進む。ステップ９０５ｈ、９０５ｉでは、次に処理終了後ステップ９０７に進む。ステップ９０５ａ、９０５ｂ、９０５ｃ、９０５ｄ、９０５ｅ、９０５ｆ、９０５ｇ、９０５ｈおよび９０５ｉでは、それぞれ命令の実行にかかるサイクル数を求める。これは、命令の種類およびその実行時のＣＰＵの状態により決定され、求まる。

【００６６】ステップ９０６では、ＣＰＵシミュレーション部２０９のプログラムカウンタレジスタの値を入力し、値４を加算して格納する。これによりプログラム実行が逐次的に行われる。ステップ９０７では、ステップ９０５ａ、９０５ｂ、９０５ｃ、９０５ｄ、９０５ｅ、９０５ｆ、９０５ｇで得られた演算結果を、ステップ９０２での解析で判別した出力先に格納する。演算結果の出力先は、命令によって異なり、レジスタやメモリ上の領域である。出力先がレジスタである場合は、ＣＰＵシミュレーション部２０９の中のレジスタ内容管理領域へ格納する。また、出力先がメモリ上の領域の場合は、メモリシミュレーション部２１０に出力先アドレスと出力データおよび出力データ長を指定することにより、出力を行う。

【００６７】ステップ９０８では、命令の読込みにかかるサイクル数（時間）、命令の実行に必要となるデータの入力にかかるサイクル数、命令の実行にかかるサイクル数、演算結果の格納にかかるサイクル数など１命令実行に必要なサイクル数を全て算出し、時刻管理部２１２

15

へ出力する。図２における時刻管理部２１２は、１命令ごとのサイクル数を合計していくことにより、実行プログラムのサイクル数を求める。

【００６８】図７にメモリシミュレーション部２１０の動作フローチャートを示す。図７において、ステップ１００１では、メモリシミュレーション部２１０へのアクセスの種別の判定を行う。前述したようにＣＰＵシミュレーション部２０９からは、命令の読込み動作（フェッチ動作）の要求、データの読込み動作（データリード動作）の要求およびデータの書きこみ動作（データライト動作）の要求がくる。また、デバッガ２０１からは、デバッガコマンドでのメモリ参照操作の場合に、データの読込み（ダイレクトリード動作）の要求が、またオブジェクトコード２１３のロード処理やデバッガ２０１上で＊

16

＊のメモリ変更操作処理の場合に、データの書きこみ（ダイレクトライト動作）の要求がくる。このダイレクトリード動作あるいは、ダイレクトライト動作の場合は、プロセッサのシミュレーションは行わない。つまりプロセッサの状態を変更せず、またサイクル数を算出しない。

【００６９】フェッチ動作の場合は、ステップ１００２で、アクセス対象として指定されたアドレスがキャッシャブル空間かどうかを判定する。この実施の形態で対象とするプロセッサは、アクセスの対象となるアドレスによってキャッシュが動作するかどうか違い、表４に示すような、区分となっている。

【００７０】

【表４】

アドレス範囲	キャッシャブル属性
0x00000000-0x3FFFFFFF	非キャッシャブル空間
0x40000000-0x7FFFFFFF	キャッシャブル空間
0x80000000-0xFFFFFFFF	非キャッシャブル空間

【００７１】表４は、メモリシミュレーション部２１０が保持するキャッシャブル属性判定表である。表４は、アドレス０ｘ００００００００番地から０ｘ３ＦＦＦＦＦＦＦ番地までのアドレス空間およびアドレス０ｘ８０００００００番地から０ｘＦＦＦＦＦＦＦＦ番地までのアドレス空間が非キャッシャブル空間であり、アドレス０ｘ４０００００００番地から０ｘ７ＦＦＦＦＦＦＦ番地までのアドレス空間がキャッシャブル空間であることを示している。キャッシャブル空間の場合、その空間内のアドレスへのアクセスは全てキャッシュメモリを利用して高速化され、非キャッシャブル空間の場合は、キャッシュメモリが利用されない。

【００７２】ステップ１００２において、指定アドレスがキャッシャブル空間の場合は、ステップ１００３に進む。そうでない場合はステップ１００４に進む。ステップ１００３では、キャッシュメモリシミュレーション部２１１に対してアドレスと読込むデータ長を指定することにより、キャッシュメモリのフェッチ動作のシミュレーションを行う。

【００７３】ステップ１００４では、メモリシミュレーション部２１０内のメモリデータ格納領域を参照し、指定アドレスのメモリデータを得る。メモリデータ格納領域は図８に示すように、実際のメモリデータとそのアドレスを管理保持する領域である。図８において、１２０１は、アドレス０ｘ００００００００番地から０ｘ０００３ＦＦＦ番地までのアドレス空間のデータ内容を保持管理する領域である。同様に、１２０２はアドレス０ｘ４０００００００番地から０ｘ４０００３ＦＦＦ番地までのアドレス空間のデータ内容を、１２０３は０ｘ８０００００００番地から０ｘ８０００３ＦＦＦ番地までのアドレス空間のデータ内容を保持管理する領域である。

【００７４】ステップ１００５では、アクセスにかかるサイクル数を算出する。キャッシャブル空間の場合は、キャッシュメモリシミュレーション部２１１で算出されたサイクル数を参照する。また非キャッシャブル空間の場合は、シミュレーション対象のメモリのウエイト数およびメモリとプロセッサ間のバス幅などの情報からサイクル数を算出する。

【００７５】同様に、データリード動作の場合は、ステップ１００６で、アクセス対象として指定されたアドレスがキャッシャブル空間かどうかを判定する。指定アドレスがキャッシャブル空間の場合は、ステップ１００７に進む。そうでない場合は、ステップ１００８に進む。ステップ１００７では、キャッシュメモリシミュレーション部２１１に対してアドレスと読込むデータ長を指定することにより、キャッシュメモリのデータリード動作のシミュレーションを行う。

【００７６】ステップ１００８では、ステップ１００４と同様に、メモリシミュレーション部２１０内のメモリデータ格納領域を参照し、指定アドレスのメモリデータを得る。ステップ１００９では、ステップ１００５と同様に、アクセスにかかるサイクル数を算出する。

【００７７】同様に、データライト動作の場合は、ステップ１０１０で、アクセス対象として指定されたアドレスがキャッシャブル空間かどうかを判定する。指定アドレスがキャッシャブル空間の場合は、ステップ１０１１に進む。そうでない場合は、ステップ１０１２に進む。ステップ１０１１では、キャッシュメモリシミュレーション部２１１に対してアドレスと書き込むデータとその

17

データ長を指定することにより、キャッシュメモリのデータライト動作のシミュレーションを行う。ステップ１０１２では、メモリシミュレーション部２１０内のメモリデータ格納領域のデータ内容を変更する。

【００７８】ステップ１０１３では、ステップ１００５と同様に、アクセスにかかるサイクル数を算出する。また、ダイレクトリード動作の場合は、ステップ１０１４において、直接メモリデータ格納領域のデータを参照する。同様に、ダイレクトライト動作の場合は、ステップ１０１５において、直接メモリデータ格納領域のデータを変更する。

【００７９】図１０に、キャッシュメモリシミュレーション部２１１のブロック図を示す。図１０において、１４０１は、キャッシュメモリシミュレーション部であり、命令キャッシュタグ情報管理部１４０２とデータキャッシュタグ情報管理部１４０３を備える。命令キャッシュタグ情報管理部１４０２は、命令キャッシュメモリのシミュレーション時つまりフェッチ動作のシミュレーション時に利用する図１１に示すタグ情報管理表を保持し、命令キャッシュメモリ１３０３上に主記憶装置１３０２上のどのアドレスのデータが登録されているかを管理する。

【００８０】同様に、データキャッシュタグ情報管理部１４０３は、データキャッシュメモリ１３０４のシミュレーション時つまりデータリード動作とデータライト動作時のシミュレーション時に利用するタグ情報管理表を保持し、データキャッシュメモリ１３０４上に登録されている主記憶装置１３０２上のアドレスを管理する。なお、この実施の形態で対象とするプロセッサでは、キャッシュデータアレイ部自体も、アドレスが割振れている。よって、キャッシュメモリ上に登録されているメモリデータは、通常のメモリデータと同様にメモリシミュレーション部２１０内のメモリデータ格納領域において管理する。

【００８１】図１１に、タグ情報管理表を示す。図１１において、エントリ１５０２はキャッシュメモリ上のアドレスに対応し、タグ１５０３は主記憶装置１３０２上のアドレスを表す。また、エントリ１５０２が有効であるか無効であるかつまり主記憶装置１３０２上のアドレスに対応しているかどうかを管理する有効フラグ１５０１を持つ。

【００８２】また、この実施の形態で対象とするプロセッサでは、エントリ１５０２と主記憶装置１３０２上のアドレスの対応付けはダイレクトマップ方式であり、キャッシュメモリ上のアドレス（エントリのアドレス）は主記憶装置１３０２でのアドレス値と３２ビット値０ｘ０００００７Ｆ０の論理積で求められる。つまりアドレス値の４ビット目から１０ビット目を取り出した値であり、１つのエントリにはタグで示されるアドレスから開始して１６バイト分が対応する。

18

【００８３】例えば、１５０４の情報はエントリ「０」が有効であり、主記憶装置１３０２上のアドレス０ｘ４０００１０００番地から０ｘ４０００１００Ｆ番地に対応することを示しており、また１５０５の情報はエントリ「２」が無効であることを示している。図１２に、キャッシュメモリシミュレーション部１４０１の動作フローチャートを示す。図１２において、ステップ１６０１で、シミュレーション動作の種別の判別を行う。動作がフェッチ動作の場合は、ステップ１６０２に進み、命令キャッシュメモリ１３０３上のエントリを指定されたアドレスから算出する。エントリ番号は、前述したようにアドレス値と３２ビット値０ｘ０００００７Ｆ０の論理積で求められる。

【００８４】ステップ１６０３では、エントリ１５０２のタグ情報および有効フラグを命令キャッシュタグ情報管理部１４０２から検索し、有効フラグを判定する。エントリ１５０２が有効な場合は、ステップ１６０４に進み、無効の場合はステップ１６０５に進む。ステップ１６０４では、取得したタグ情報と指定アドレスを比較し、キャッシュヒットかどうかを判定する。この比較には、指定アドレスのＬＳＢ４ビットを０でマスクした値を用いる。比較結果が一致であれば、キャッシュヒットである。一致しなければキャッシュミスである。また、ステップ１６０３でエントリが無効である場合も、キャッシュミスである。ステップ１６０４でキャッシュミスと判定された場合は、競合性ミスであり、この場合競合性ミスであることを記憶する。なお、競合性ミスであることはステップ１６０７において、キャッシュアクセス情報採取部２０７に登録される。

【００８５】キャッシュミスの場合、ステップ１６０５に進み、エントリ１５０２に登録する。ステップ１６０５では、エントリ１５０２にアドレスを登録する。ステップ１６０６では、命令キャッシュミス時のサイクル数を求める。ステップ１６０７では、命令キャッシュへのフェッチ動作時のアクセスがキャッシュミスであったこと、ならびにアクセスアドレス、データ長、エントリおよび競合性ミスであったかどうかをキャッシュアクセス情報採取部２０７に登録する。

【００８６】ステップ１６０８では、命令キャッシュヒット時のサイクル数を求める。ステップ１６０９では、命令キャッシュへのフェッチ動作時のアクセスがキャッシュヒットであっとこと、ならびにアクセスアドレス、データ長、およびエントリをキャッシュアクセス情報採取部２０７に登録する。同様に、動作がデータリード動作の場合は、ステップ１６１０に進み、データキャッシュメモリ上のエントリを指定されたアドレスから算出する。ステップ１６１１では、エントリのタグ情報および有効フラグをデータキャッシュタグ情報管理部１４０３から検索し、有効フラグを判定する。エントリが有効な場合は、ステップ１６１２に進み、無効の場合はステッ

プ１６１３に進む。

【００８７】ステップ１６１２では、取得したタグ情報と指定アドレスを比較し、キャッシュヒットかどうかを判定する。キャッシュミスと判定された場合は、競合性ミスであり、この場合競合性ミスであることを記憶する。なお、競合性ミスであることはステップ１６１５において、キャッシュアクセス情報採取手段であるキャッシュアクセス情報採取部２０７に登録される。

【００８８】キャッシュミスの場合、ステップ１６１３に進む。ステップ１６１３では、エントリにそのアドレスを登録する。ステップ１６１４では、データリード動作におけるデータキャッシュミス時のサイクル数を求める。ステップ１６１５では、データキャッシュへのデータリード動作時のアクセスがキャッシュミスであったこと、ならびにアクセスアドレス、データ長、およびエントリをキャッシュアクセス情報採取部２０７に登録する。また、ＣＰＵシミュレーション部２０９からプログラムカウンタレジスタ（ＰＣ）の値と、スタックポインタレジスタ（ＳＰ）の値を取得し、共に登録する。また、競合性ミスであったかどうも登録する。

【００８９】ステップ１６１６では、データリード動作におけるデータキャッシュヒット時のサイクル数を求める。ステップ１６１７では、データキャッシュへのデータリード動作時のアクセスがキャッシュヒットであったことをキャッシュアクセス情報採取部２０７に登録する。またアクセスアドレス、データ長、およびエントリをキャッシュアクセス情報採取部２０７に登録する。また、ＣＰＵシミュレーション部２０９からプログラムカウンタレジスタ（ＰＣ）の値と、スタックポインタレジスタ（ＳＰ）の値を取得し、共に登録する。

【００９０】同様に、動作がデータライト動作の場合は、ステップ１６１８に進み、データキャッシュメモリ上のエントリを指定されたアドレスから算出する。ステップ１６１９では、エントリのタグ情報および有効フラグをデータキャッシュタグ情報管理部１４０３から検索し、有効フラグを判定する。そのエントリが有効な場合は、ステップ１６２０に進み、無効の場合はステップ１６２１に進む。

【００９１】ステップ１６２０では、取得したタグ情報と指定アドレスを比較し、キャッシュヒットかどうかを判定する。キャッシュミスと判定された場合は、競合性ミスであり、この場合競合性ミスであることを記憶する。なお、競合性ミスであることはステップ１６２３において、キャッシュアクセス情報採取部２０７に登録される。

【００９２】キャッシュミスの場合、ステップ１６２１に進む。ステップ１６２１では、エントリにそのアドレスを登録する。ステップ１６２２では、データライト動作におけるデータキャッシュミス時のサイクル数を求める。ステップ１６２３では、データキャッシュへのデータライト動作時のアクセスがキャッシュミスであったこと、ならびにアクセスアドレス、データ長、およびエントリをキャッシュアクセス情報採取部２０７に登録する。また、ＣＰＵシミュレーション部２０９からプログラムカウンタレジスタ（ＰＣ）の値と、スタックポインタレジスタ（ＳＰ）の値を取得し、共に登録する。また、競合性ミスであったかどうも登録する。

【００９３】ステップ１６２４では、データライト動作におけるデータキャッシュヒット時のサイクル数を求める。ステップ１６２５では、データキャッシュへのデータライト動作時のアクセスがキャッシュヒットであっとことをキャッシュアクセス情報採取部２０７に登録する。また、アクセスアドレス、データ長、およびエントリをキャッシュアクセス情報採取部２０７に登録する。また、ＣＰＵシミュレーション部２０９からプログラムカウンタレジスタ（ＰＣ）の値と、スタックポインタレジスタ（ＳＰ）の値を取得し、共に登録する。

【００９４】図１３に、キャッシュアクセス情報採取部２０７のブロック図を示す。図１３において、１７０１で示すキャッシュアクセス情報採取部２０７は、命令キャッシュアクセス情報管理部１７０２と、データキャッシュアクセス情報管理部１７０３を備える。命令キャッシュアクセス情報管理部１７０２は、フェッチアクセス情報１７０４を保持する。フェッチアクセス情報１７０４は、キャッシュメモリでのエントリ番号、フェッチアクセスにおけるアクセスアドレスとそのアクセスがキャッシュヒットであったかどうかの情報、そのデータ長、また対応するシンボルとそのアクセスの回数およびキャッシュミスの場合はその内の競合性ミスの回数を含む情報である。

【００９５】また同様に、データキャッシュアクセス情報管理部１７０３は、データリードアクセス情報１７０５とデータライトアクセス情報１７０６を保持する。データリードアクセス情報１７０５は、データリードアクセスにおけるアクセスアドレスとそのアクセスがキャッシュヒットであったかどうかの情報、そのデータ長、また対応するシンボルとそのアクセスの回数、およびキャッシュミスの場合はその内の競合性ミスの回数を含む情報である。

【００９６】同様に、データライトアクセス情報１７０６は、データライトアクセスにおけるアクセスアドレスとそのアクセスがキャッシュヒットであったかどうかの情報、そのデータ長、また対応するシンボルとそのアクセスの回数、およびキャッシュミスの場合はその内の競合性ミスの回数を含む情報である。図１４に、キャッシュアクセス情報採取部２０７の動作フローチャートを示す。図１４において、ステップ１８０１では、キャッシュメモリシミュレーション部１４０１から入力された採取情報の種別の判定を行う。採取情報が、フェッチ動作に関するものならばステップ１８０２に進み、データリ

ード動作に関するものならばステップ１８０４に進み、データライト動作に関するものならばステップ１８０６に進む。

【００９７】ステップ１８０２では、フェッチアクセスの対象アドレスをシンボル情報から検索し、シンボルとの関連を付ける。すなわち、アドレスをキーにシンボル情報を検索し、シンボル名を得る。ステップ１８０３では、フェッチアクセス情報１７０４に、アクセス対象のアドレス、キャッシュヒットか否か、アクセスデータ長、対応付けられたシンボルを登録する。既に同一の情報が登録されていればそのアクセスの回数の値を増やす。

【００９８】ステップ１８０４では、データリードアクセスの対象アドレスをシンボル情報から検索し、シンボルとの関連を付ける。すなわち、アドレスをキーにシンボル情報を検索し、シンボル名を得る。また、その際にシンボル情報中で局所変数のように関数とスタックポインタレジスタ（ＳＰ）からのオフセットして定義されているシンボル情報は、前記の採取情報に含まれているアクセス時のプログラムカウンタ（ＰＣ）から関数が判別でき、さらにアクセス時のスタックポインタレジスタ（ＳＰ）の値からシンボルとの対応付けができる。

【００９９】ステップ１８０５では、データリードアクセス情報１７０５に、アクセス対象のアドレス、キャッシュヒットか否か、アクセスデータ長、対応付けられたシンボルを登録する。既に同一の情報が登録されていればそのアクセスの回数の値を増やす。同様に、ステップ１８０６では、データライトアクセスの対象アドレスをシンボル情報から検索し、シンボルとの関連を付ける。すなわち、アドレスをキーにシンボル情報を検索し、シンボル名を得る。また、その際にシンボル情報中で局所変数のように関数とスタックポインタレジスタ（ＳＰ）からのオフセットして定義されているシンボル情報は、前記の採取情報に含まれているアクセス時のプログラム＊カウンタ（ＰＣ）から関数が判別でき、さらにアクセス時のスタックポインタレジスタ（ＳＰ）の値からシンボルとの対応付けができる。

【０１００】ステップ１８０７では、データライトアクセス情報１７０６に、アクセス対象のアドレス、キャッシュヒットか否か、アクセスデータ長、および、対応付けられたシンボルを登録する。既に同一の情報が登録されていればそのアクセスの回数の値を増やす。なお、シンボル名が対応付かない場合は、シンボル名なしとして登録する。

【０１０１】図１５に、プログラムのシンボルの配置を変更するためのシンボル配置変更情報を生成するシンボル配置変更情報生成部２０８の動作フローチャートを示す。ステップ１９０１において、命令キャッシュに関する情報であるフェッチアクセス情報１７０４について、各エントリの情報を走査し、競合性ミスが発生しなかったシンボル、競合性ミスが発生したシンボルをそれぞれ図１６（ａ）に示す命令キャッシュ競合情報表２１０１に登録する。

【０１０２】ステップ１９０２において、命令キャッシュ競合情報管理表２１０１から競合ミスをおこしたシンボルを検索する。シンボルがあればステップ１９０３に進み、シンボルがなければステップ１９０５に進む。ステップ１９０３において、命令キャッシュ競合情報管理表２１０１からキャッシュがヒットしていない空のエントリを検索する。空のエントリがあればステップ１９０４に進み、空のエントリがなければステップ１９０５に進む。

【０１０３】ステップ１９０４において、競合ミスをおこしたシンボルを空のエントリ割付け、対応するアドレスを生成する。空のエントリ番号からアドレスの変更は、例えばある既定値からのエントリ番号に対応するオフセットを足し込むなどして求めることが可能である。生成する情報であるシンボル配置変更情報は、たとえば

変数ｘ：０ｘ０４００　→　０ｘ１１４０　　…（1）

のように表示する。

【０１０４】データキャッシュに関して同様に、ステップ１９０５から１９０８まで、それぞれステップ１９０１から１９０４に対応する処理を行う。ステップ１９０５において、ステップ１９０１と同様に、データキャッシュに関する情報であるデータリードアクセス情報１７０５およびデータライトアクセス情報１７０６について、各エントリの情報を走査し、図１６（ｂ）に示すデータキャッシュ競合情報表２１０２に登録する。ステップ１９０６において、データキャッシュ競合情報管理表２１０２から競合ミスをおこしたシンボルを検索する。シンボルがあればステップ１９０７に進み、シンボルがなければ処理を終了する。例えば、競合するシンボルとしては、『変数ｘ』と『変数ｉ＠ｍａｉｎ』が得られる。

【０１０５】ステップ１９０７において、データキャッシュ競合情報管理表２１０２からキャッシュがヒットしていない空のエントリを検索する。空のエントリがあればステップ１９０８に進み、空のエントリがなければ処理を終了する。例えば、空のエントリとして０ｘ１４が得られる。ステップ１９０８において、競合ミスをおこしたシンボルを空のエントリ割付け、対応するアドレスを生成する。

【０１０６】上記の表示（1）に、シンボル配置変更情報生成部２０８によるシンボル配置変更情報の一例を示す。表示（1）は、シンボルｘをアドレス０ｘ０４００から０ｘ１１４０へ変更することを示している。以上のように、この第１の実施の形態によれば、キャッシュメモリへのアクセスの状況を高級言語のシンボルとして検出し、キャッシュメモリで競合性ミスが発生するシンボ

ルの配置を変更する情報が生成できる。

【０１０７】この発明の第２の実施の形態を図１７から図１９により説明する。すなわち、このソフトウエア開発支援システムは、第１の実施の形態の図２に示すシンボル配置変更情報生成部２０８に代えて、表示手段であるキャッシュアクセス情報表示部２１６を設けたものである。図１８に、キャッシュアクセス情報表示部２１６の動作フローチャートを示す。ステップ１９０１において、命令キャッシュに関する情報であるフェッチアクセス情報をエントリ別にわけて表示する。ステップ１９０２において、データキャッシュに関する情報であるデータリードアクセス情報とデータライトアクセス情報とをエントリ別にわけて表示する。

【０１０８】図１９にキャッシュアクセス情報表示部２１６によるキャッシュアクセス情報の一例を示す。図１９において、エントリおよびシンボル別にデータリード動作、データライト動作のヒットとミスの頻度の表示を示している。このように、キャッシュメモリへのアクセスの状況を高級言語レベルで利用者に表示することができる。

【０１０９】第２の実施の形態によれば、キャッシュアクセス情報採取部２０７で採取された情報と、シンボル情報管理部２０６で管理されているシンボル情報をキャッシュアクセス情報表示部２１６により関連付けて表示することができるので、キャッシュメモリへのアクセスの状況が、対象プログラムが記述される高級言語レベルで表示可能となり、評価性能に基いたプログラムの変更が容易になる。したがって、キャッシュメモリ搭載のプログラムを開発し評価する際に、キャッシュメモリへのアクセス状況を高級言語レベルで知ることができ、プログラム開発の効率化が図れる。

【０１１０】その他は、第１の実施の形態と同様である。なお、他の実施の形態として、第１の実施の形態に第２の実施の形態のキャッシュアクセス情報表示部２１６を付加したソフトウエア開発支援システムも可能である。また、この発明において、キャッシュアクセス情報を採取する区間を指定する手段を設け、指定された区間についてのみキャッシュアクセスに関する情報を採取することにより、高速性が求められるプログラムの特定の区間のみについて、性能向上を図ることも可能である。

【０１１１】

【発明の効果】請求項１記載のソフトウエア開発支援システムによれば、キャッシュアクセス情報採取手段で採取された情報と、シンボル情報管理手段で管理されているシンボル情報から、プログラムのメモリアクセス時の速度が高速になるようにシンボルの配置を変更するためのシンボルの配置変更情報を生成することができる。これにより、キャッシュメモリへのアクセスの状況が対象プログラムが記述される高級言語レベルで検知可能となり、アクセス速度を低下させるシンボルの配置を変更

し、プログラムの実行性能を向上させることが可能となる。

【０１１２】請求項２記載のソフトウエア支援開発システムによれば、請求項１と同様な効果のほか、キャッシュアクセス情報採取手段で採取された情報からキャッシュ競合性ミスを検出し、シンボル情報と関連づけて、シンボルの配置変更情報をシンボル配置変更情報生成手段で生成することができる。これにより、キャッシュメモリ搭載のプログラムを開発する際に、キャッシュメモリへのアクセスの状況特に競合性ミスをシンボルレベルないし高級言語レベルで検出し、キャッシュメモリで競合性ミスが発生するシンボルの配置を変更する情報を生成するなど、その競合を解消するためにシンボルの配置を変更することが容易にでき、プログラム開発の効率化が図れる。

【０１１３】請求項３記載のソフトウエア支援開発システムによれば、請求項１または請求項２と同様な効果のほか、競合性のキャッシュミスが発生したシンボルをキャッシュされていない空のエントリに対応するアドレスに変更するための情報を生成するなど、キャッシュメモリの競合性ミスを低減させるようなシンボルの配置変更情報が生成でき、より一層のプログラムの性能向上を図れる。

【０１１４】請求項４記載のソフトウエア支援開発システムによれば、キャッシュアクセス情報採取手段で採取された情報と、シンボル情報管理手段で管理されているシンボル情報を表示手段により関連付けて表示することができるので、キャッシュメモリへのアクセスの状況が、対象プログラムが記述される高級言語レベルで表示可能となり、評価性能に基いたプログラムの変更が容易になる。したがって、キャッシュメモリ搭載のプログラムを開発し評価する際に、キャッシュメモリへのアクセス状況を高級言語レベルで知ることができ、プログラム開発の効率化が図れる。

【０１１５】請求項５記載のソフトウエア支援開発システムによれば、請求項４と同様な効果がある。請求項６記載のソフトウエア支援開発システムによれば、請求項４または請求項５と同様な効果がある。

【図面の簡単な説明】

【図１】この発明の第１の実施の形態におけるソフトウエア開発支援システムに適用されるシステムのブロック図である。

【図２】この発明の第１の実施の形態におけるソフトウエア開発支援システムのブロック図である。

【図３】キャッシュメモリ搭載プロセッサのシステムブロック図である。

【図４】プロセッサの動作モデルを説明する説明図である。

【図５】図２の実行制御部２０４の動作フローチャートである。

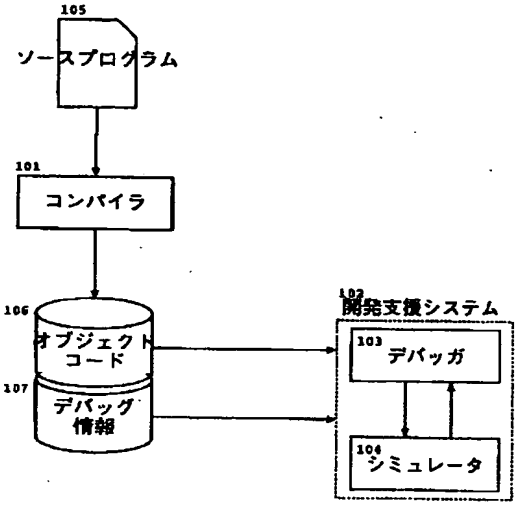
25

【図６】図２のＣＰＵシミュレーション部２０９の動作フローチャートである。
【図７】図２のメモリシミュレーション部２１０の動作フローチャートである。
【図８】メモリシミュレーション部のメモリデータ格納領域の説明図である。
【図９】この実施の形態におけるキャッシュメモリ搭載プロセッサのシステムブロック図である。
【図１０】キャッシュシミュレーション部のブロック図である。
【図１１】キャッシュタグ情報管理表を示す説明図である。
【図１２】キャッシュシミュレーション部２１１の動作フローチャートである。
【図１３】キャッシュアクセス情報採取部のブロック図である。
【図１４】キャッシュアクセス情報採取部の動作フローチャートである。
【図１５】シンボル配置変更情報生成部の動作フローチャートである。
【図１６】キャッシュ競合情報表を示すブロック図である。
【図１７】第２の実施の形態におけるソフトウエア開発支援システムのブロック図である。
【図１８】キャッシュアクセス情報表示部の動作フローチャートである。
【図１９】キャッシュアクセス情報の表示例を示す説明図である。
【符号の説明】
１０１　コンパイラ
１０２　開発支援システム
１０３　デバッガ
１０４　シミュレータ
１０５　ソースプログラム
１０６　オブジェクトコード
１０７　デバッグ情報
２０１　デバッガ
２０２　シミュレータ
２０３　デバッグ操作処理部
２０４　実行制御部
２０５　オブジェクトコード管理部
２０６　シンボル情報管理部
２０７　キャッシュアクセス情報採取部
２０８　シンボル配置変更情報生成部
２０９　ＣＰＵシミュレーション部
２１０　メモリシミュレーション部

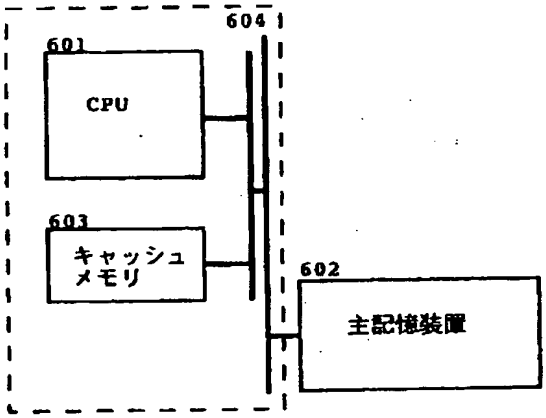
26

２１１　キャッシュメモリシミュレーション部
２１２　時刻管理部
２１６　キャッシュアクセス情報表示部
６０１　ＣＰＵ
６０２　主記憶装置
６０３　キャッシュメモリ
６０４　バス
７０１　ＣＰＵ
７０２　メモリ
７０３、７０４、７０５、７０６　データレジスタ
７０７、７０８、７０９、７１０　アドレスレジスタ
７１１　プロセッサ状態語レジスタ
７１２　プログラムカウンタレジスタ
７１３　スタックポインタレジスタ
７１４　演算器
７１５　実行コード格納領域
７１６　データ格納領域
７１７　スタック領域
１２０１、１２０２、１２０３　アドレス空間メモリデータ格納領域
１３０１　ＣＰＵ
１３０２　主記憶装置
１３０３　命令キャッシュメモリ
１３０４　データキャッシュメモリ
１３０５　バス
１３０６　命令キャッシュメモリのタグアレイ
１３０７　命令キャッシュメモリのデータアレイ
１３０８　データキャッシュメモリのタグアレイ
１３０９　データキャッシュメモリのデータアレイ
１４０１　キャッシュメモリシミュレーション部
１４０２　命令キャッシュタグ情報管理部
１４０３　データキャッシュタグ情報管理部
１５０１　有効フラグ
１５０２　エントリ
１５０３　タグ
１５０４　有効なエントリの一例
１５０５　無効なエントリの一例
１７０１　キャッシュアクセス情報採取部
１７０２　命令キャッシュアクセス情報管理部
１７０３　データキャッシュアクセス情報管理部
１７０４　フェッチアクセス情報
１７０５　データリードアクセス情報
１７０６　データライトアクセス情報
２１０１　命令キャッシュ競合情報管理表
２１０２　データキャッシュ競合情報管理表

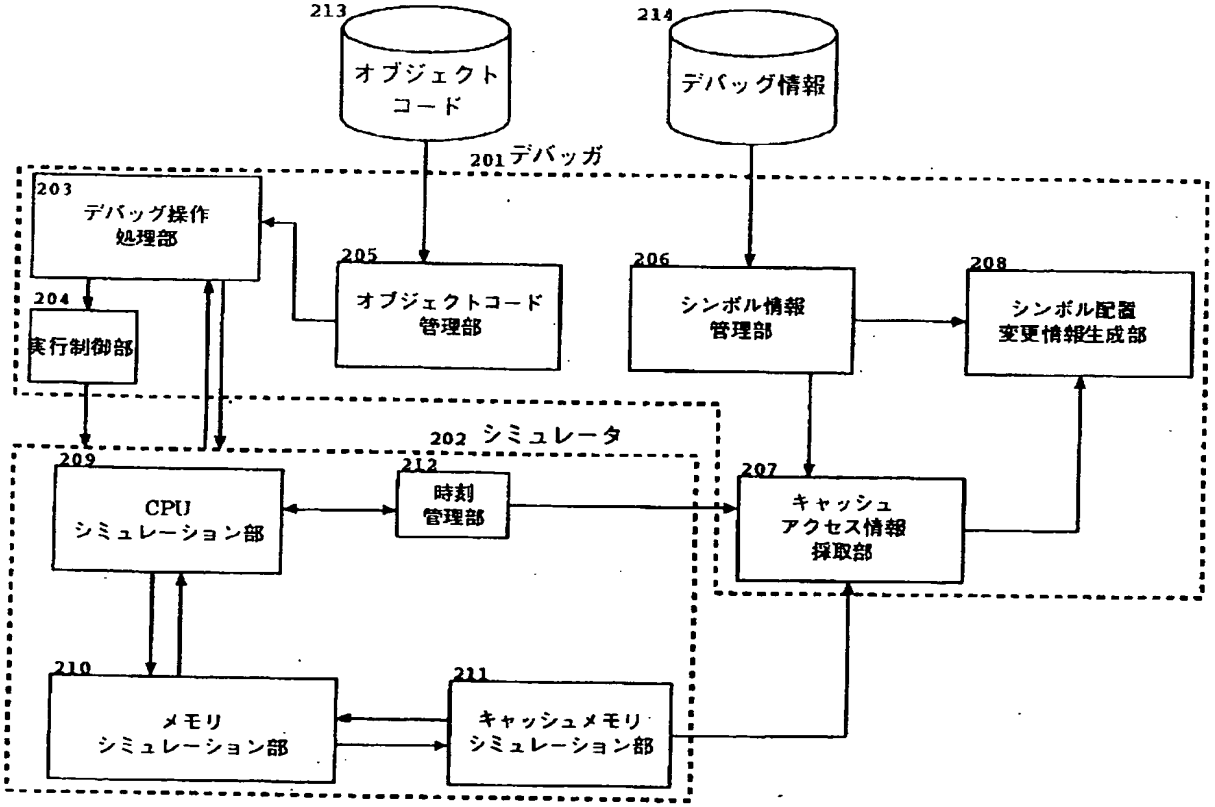
【図１】


105
ソースプログラム
101
コンパイラ
106
オブジェクトコード
107
デバッグ情報
102
開発支援システム
103
デバッガ
104
シミュレータ

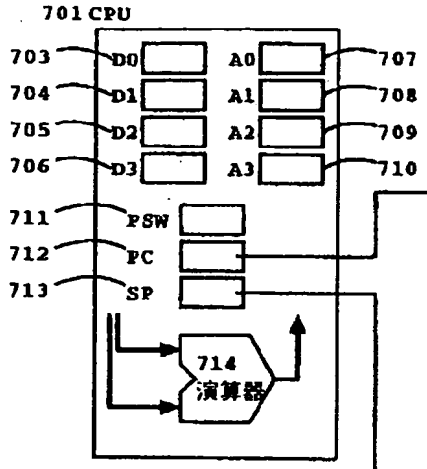
【図３】


604
601
CPU
603
キャッシュメモリ
602
主記憶装置

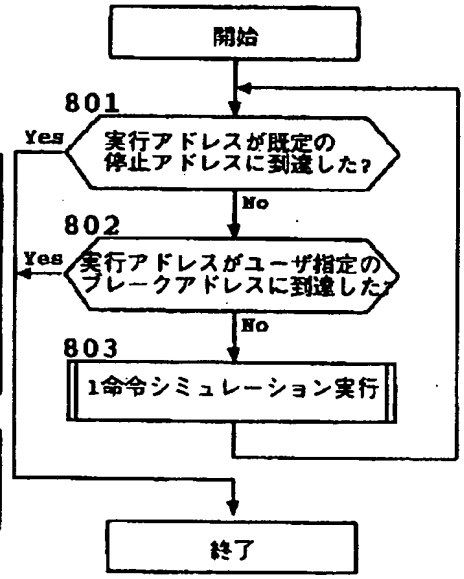
【図２】

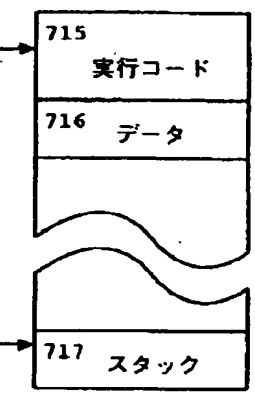

213
オブジェクトコード
214
デバッグ情報
201デバッガ
203
デバッグ操作処理部
204
実行制御部
205
オブジェクトコード管理部
206
シンボル情報管理部
208
シンボル配置変更情報生成部
202 シミュレータ
209
CPUシミュレーション部
212
時刻管理部
207
キャッシュアクセス情報採取部
210
メモリシミュレーション部
211
キャッシュメモリシミュレーション部

【図４】

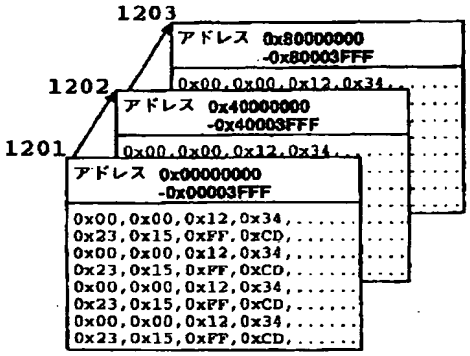


【図５】

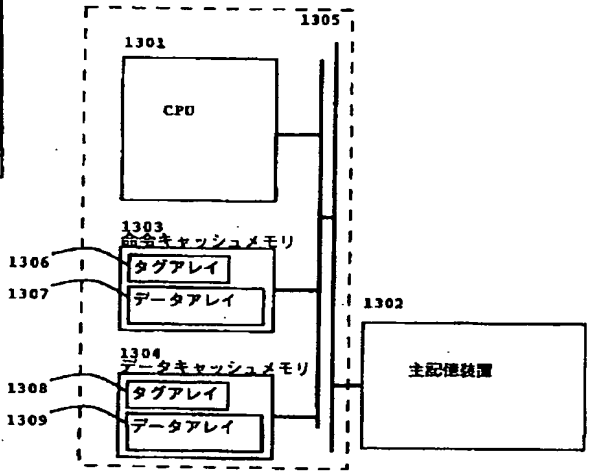




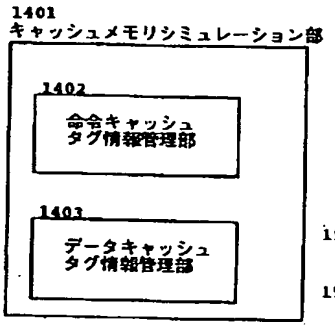
【図８】



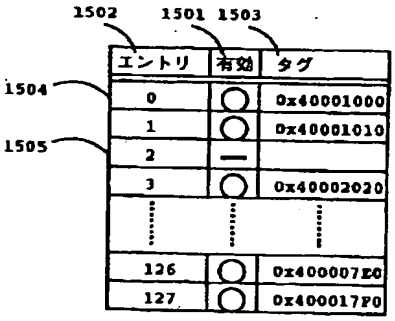
【図９】



【図１０】



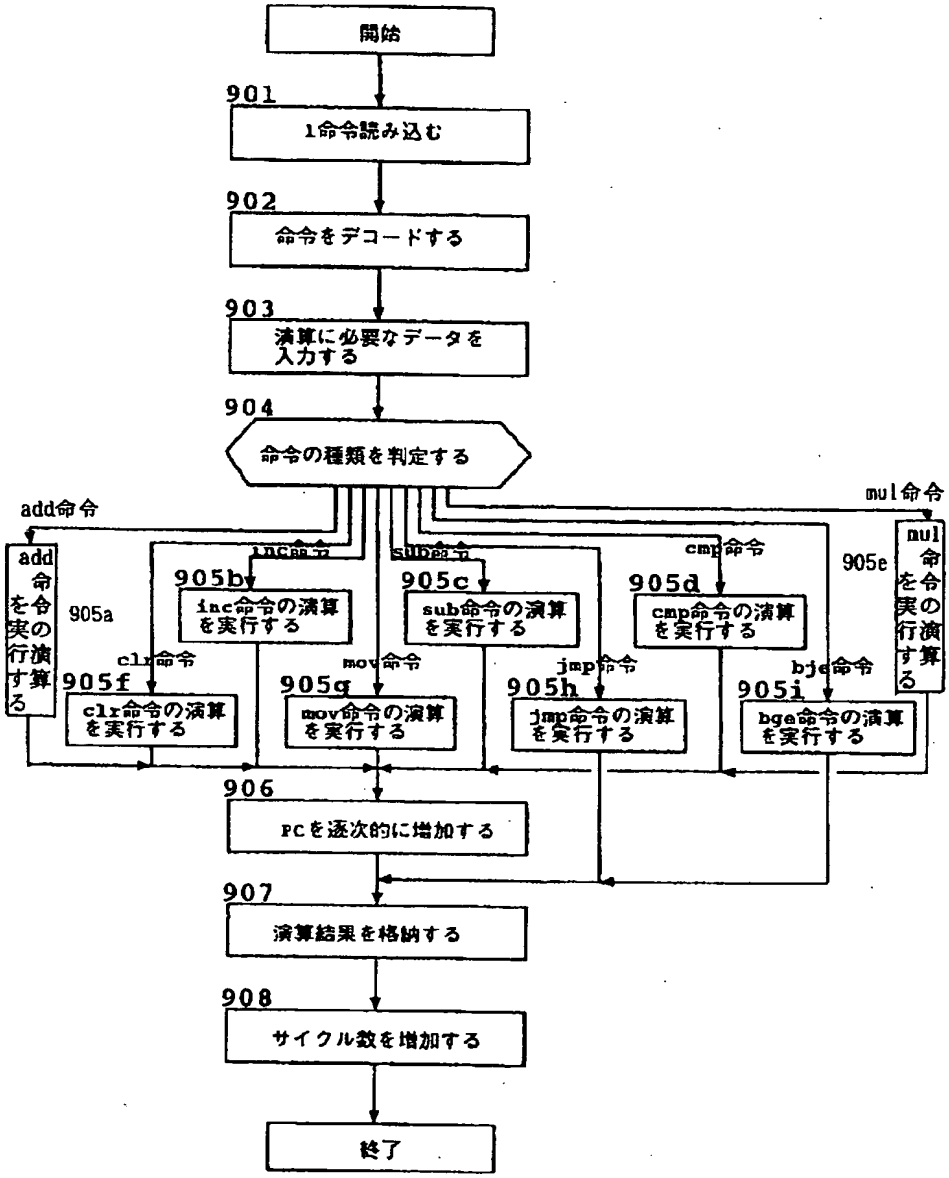
【図１１】



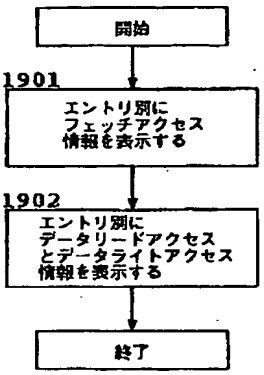
【図１９】

エントリ	シンボル	ヒット		ミス	
		リード	ライト	リード	ライト
0x40	変数x	0	0	0	101
	変数i@main	300	100	100	1
0x41	変数y	100	100	0	1

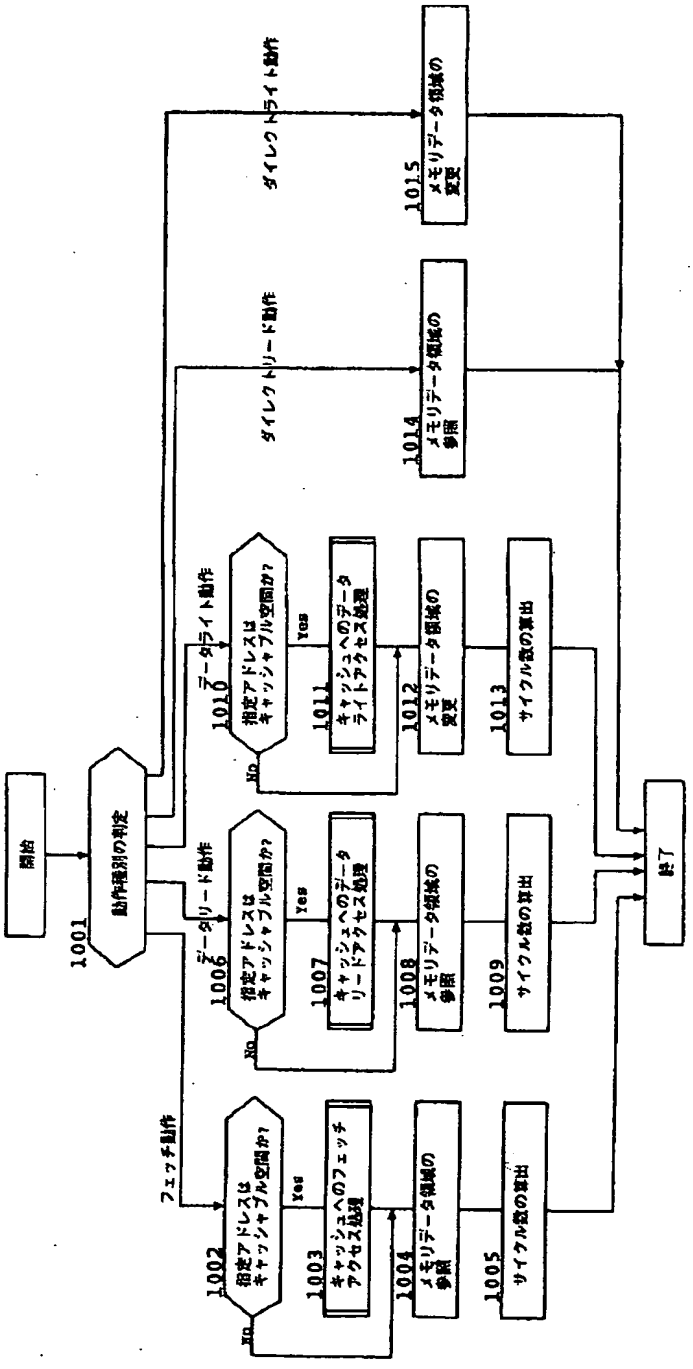
【図６】


開始
901
1命令読み込む
902
命令をデコードする
903
演算に必要なデータを入力する
904
命令の種類を判定する
add命令
mul命令
add命令の演算を実行する
905a
incの命令
905b
inc命令の演算を実行する
subの命令
905c
sub命令の演算を実行する
cmp命令
905d
cmp命令の演算を実行する
905e
mul命令の演算を実行する
clr命令
905f
clr命令の演算を実行する
mov命令
905g
mov命令の演算を実行する
jmp命令
905h
jmp命令の演算を実行する
bje命令
905i
bge命令の演算を実行する
906
PCを逐次的に増加する
907
演算結果を格納する
908
サイクル数を増加する
終了

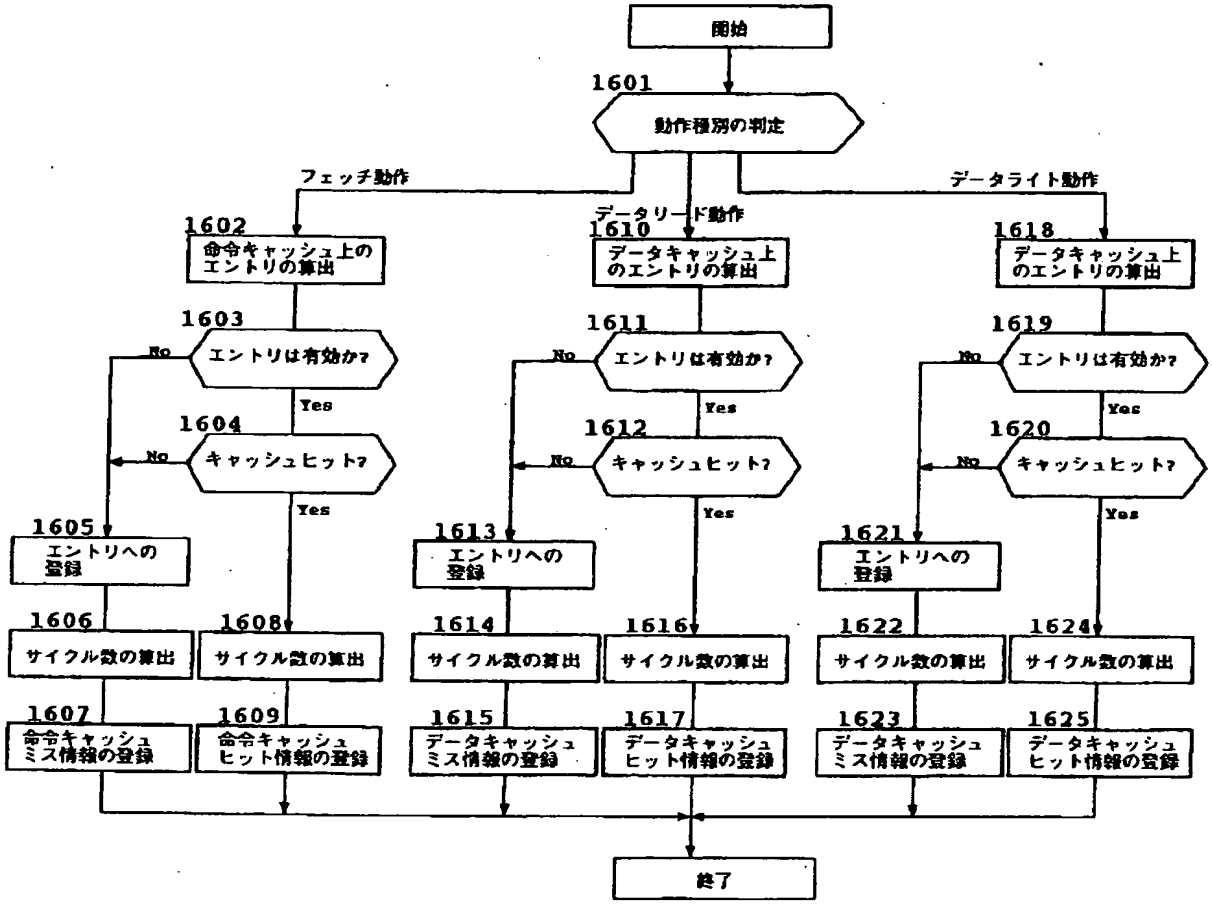
【図１８】


開始
1901
エントリ別にフェッチアクセス情報を表示する
1902
エントリ別にデータリードアクセスとデータライトアクセス情報を表示する
終了

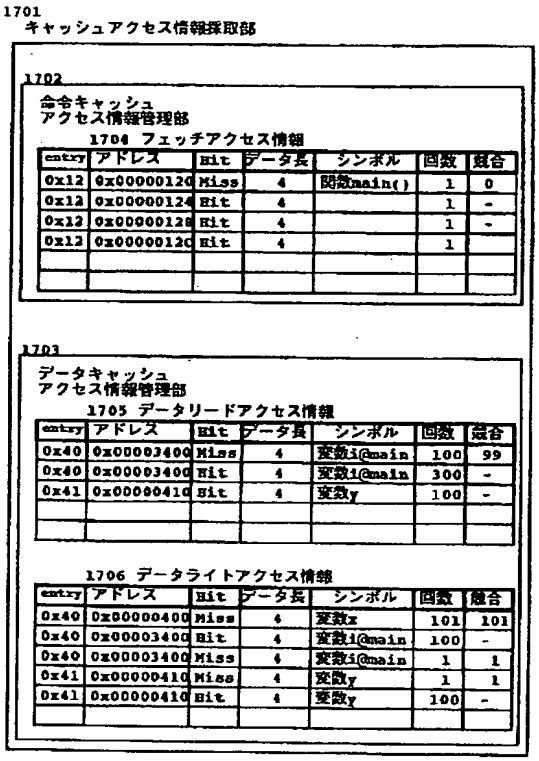
【図7】


開始
動作種別の判定
1001
フェッチ動作
1002
指定アドレスは
キャッシュブル空間か?
No
Yes
1003
キャッシュへのフェッチ
アクセス処理
1004
メモリデータ領域の
参照
1005
サイクル数の算出
データリード動作
1006
指定アドレスは
キャッシュブル空間か?
No
Yes
1007
キャッシュへのデータ
リードアクセス処理
1008
メモリデータ領域の
参照
1009
サイクル数の算出
データライト動作
1010
指定アドレスは
キャッシュブル空間か?
No
Yes
1011
キャッシュへのデータ
ライトアクセス処理
1012
メモリデータ領域の
変更
1013
サイクル数の算出
ダイレクトリード動作
1014
メモリデータ領域の
参照
ダイレクトライト動作
1015
メモリデータ領域の
変更
終了

【図１２】


開始
1601
動作種別の判定
フェッチ動作
データリード動作
データライト動作
1602
命令キャッシュ上のエントリの算出
1603
No
エントリは有効か?
Yes
1604
No
キャッシュヒット?
Yes
1605
エントリへの登録
1606
サイクル数の算出
1607
命令キャッシュミス情報の登録
1608
サイクル数の算出
1609
命令キャッシュヒット情報の登録
1610
データキャッシュ上のエントリの算出
1611
No
エントリは有効か?
Yes
1612
No
キャッシュヒット?
Yes
1613
エントリへの登録
1614
サイクル数の算出
1615
データキャッシュミス情報の登録
1616
サイクル数の算出
1617
データキャッシュヒット情報の登録
1618
データキャッシュ上のエントリの算出
1619
No
エントリは有効か?
Yes
1620
No
キャッシュヒット?
Yes
1621
エントリへの登録
1622
サイクル数の算出
1623
データキャッシュミス情報の登録
1624
サイクル数の算出
1625
データキャッシュヒット情報の登録
終了

【図１３】



1704 フェッチアクセス情報

entry	アドレス	Hit	データ長	シンボル	回数	競合
0x12	0x00000120	Miss	4	関数main()	1	0
0x12	0x00000124	Hit	4		1	-
0x12	0x00000128	Hit	4		1	-
0x12	0x0000012C	Hit	4		1	

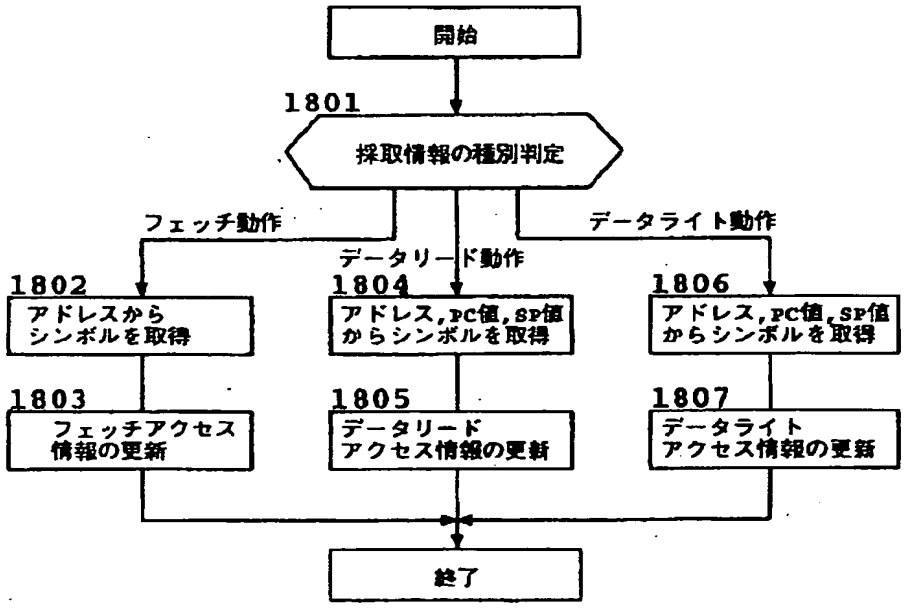
1705 データリードアクセス情報

entry	アドレス	Hit	データ長	シンボル	回数	競合
0x40	0x00003400	Miss	4	変数i@main	100	99
0x40	0x00003400	Hit	4	変数i@main	300	-
0x41	0x00000410	Hit	4	変数y	100	-

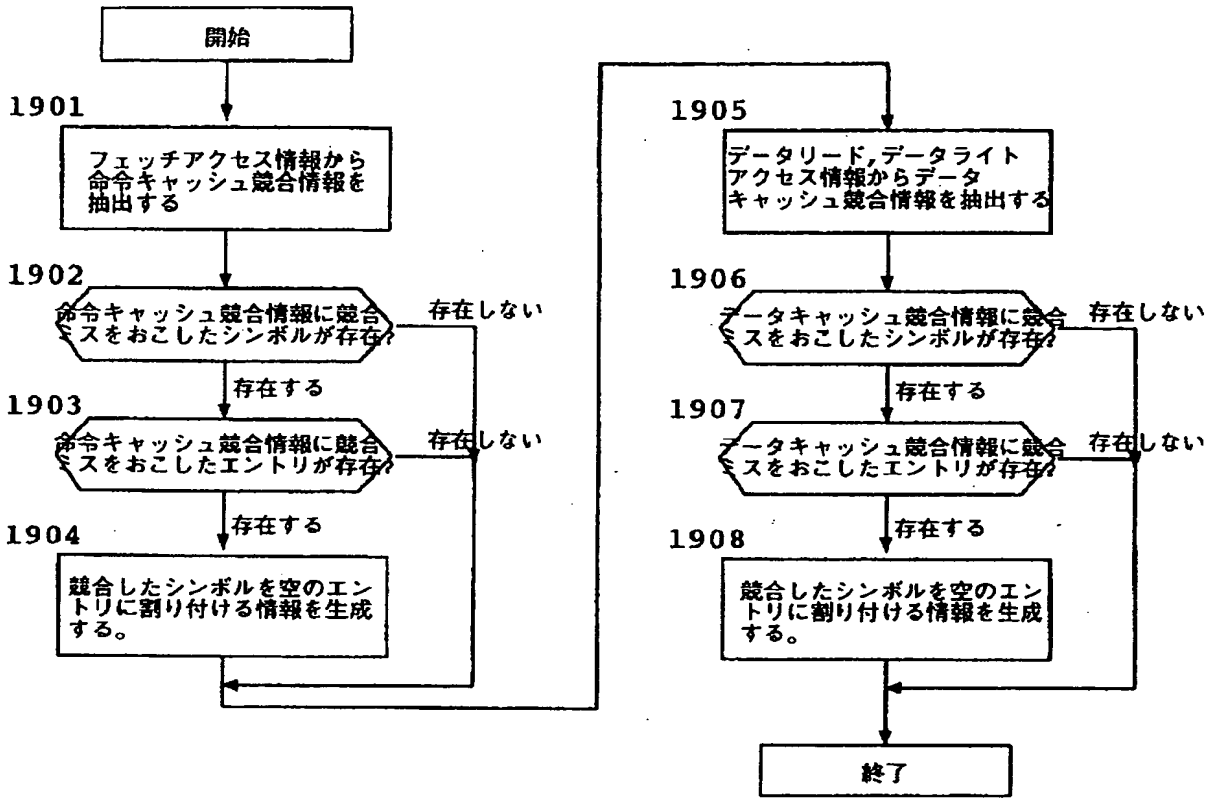
1706 データライトアクセス情報

entry	アドレス	Hit	データ長	シンボル	回数	競合
0x40	0x00000400	Miss	4	変数x	101	101
0x40	0x00003400	Hit	4	変数i@main	100	-
0x40	0x00003400	Miss	4	変数i@main	1	1
0x41	0x00000410	Miss	4	変数y	1	1
0x41	0x00000410	Hit	4	変数y	100	-

【図１４】



【図１５】


開始
1901
フェッチアクセス情報から命令キャッシュ競合情報を抽出する
1902
命令キャッシュ競合情報に競合ミスをおこしたシンボルが存在?
存在しない
存在する
1903
命令キャッシュ競合情報に競合ミスをおこしたエントリが存在?
存在しない
存在する
1904
競合したシンボルを空のエントリに割り付ける情報を生成する。
1905
データリード,データライトアクセス情報からデータキャッシュ競合情報を抽出する
1906
データキャッシュ競合情報に競合ミスをおこしたシンボルが存在?
存在しない
存在する
1907
データキャッシュ競合情報に競合ミスをおこしたエントリが存在?
存在しない
存在する
1908
競合したシンボルを空のエントリに割り付ける情報を生成する。
終了

【図１６】

(a)

2101　命令キャッシュ競合情報管理表

1706

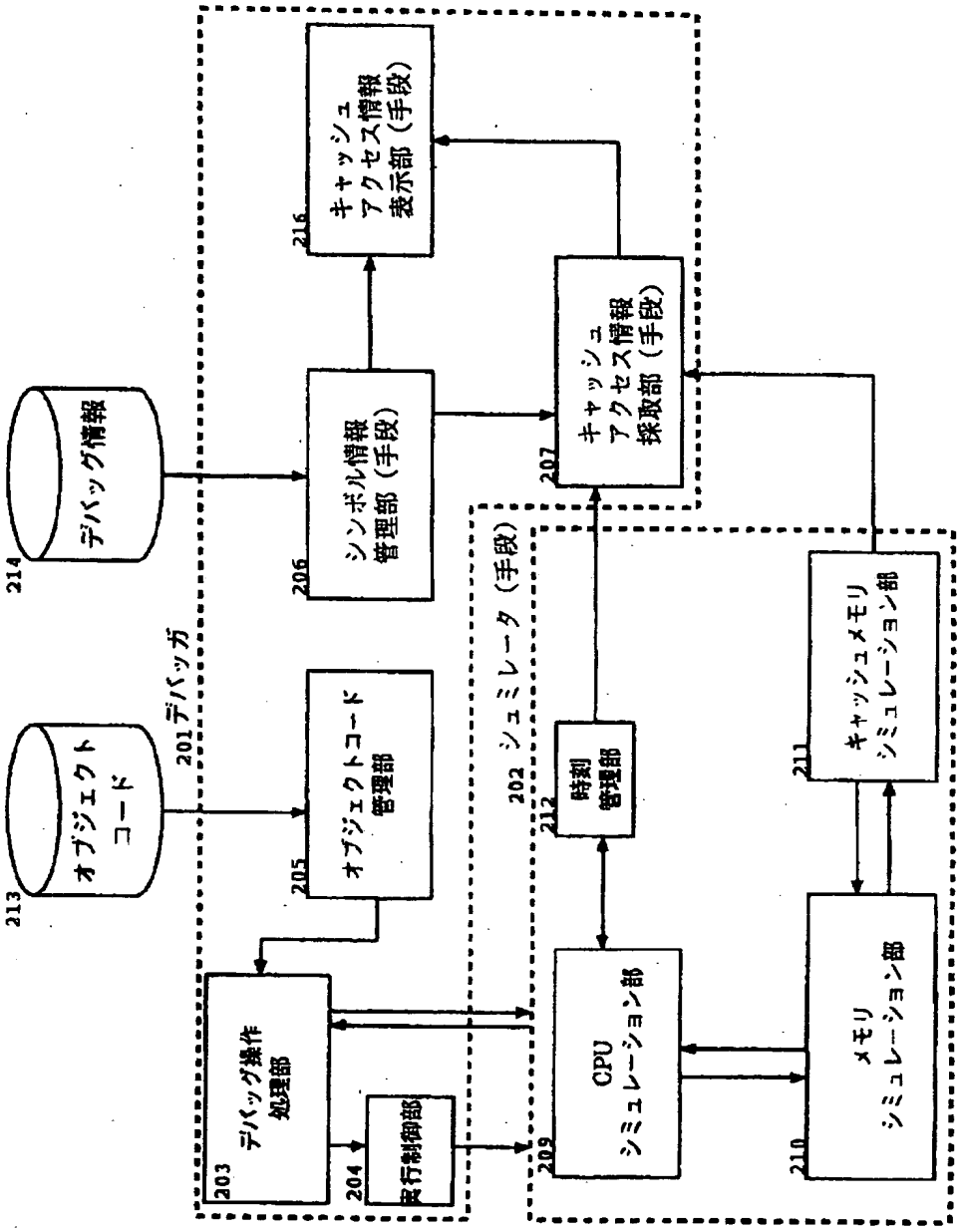
entry	シンボル	競合ミスをおこしたシンボル
0x00		
0x01		
⋮		
0x10		
0x11		
0x12	関数main()	
⋮		
0x7E		
0x7F		

(b)

2102　データキャッシュ競合情報管理表

entry	シンボル	競合ミスをおこしたシンボル
0x00		
0x01		
⋮		
0x40		変数x，変数i@main
0x41	変数y	
0x42		
⋮		
0x7E		
0x7F		

【図１７】


デバッグ情報
214
オブジェクトコード
213
201 デバッガ
シンボル情報管理部（手段）
206
キャッシュアクセス情報表示部（手段）
216
キャッシュアクセス情報採取部（手段）
207
オブジェクトコード管理部
205
202 シミュレータ（手段）
時刻管理部
212
キャッシュメモリシミュレーション部
211
デバッグ操作処理部
203
実行制御部
204
CPUシミュレーション部
209
メモリシミュレーション部
210